**Android activity启动过程**

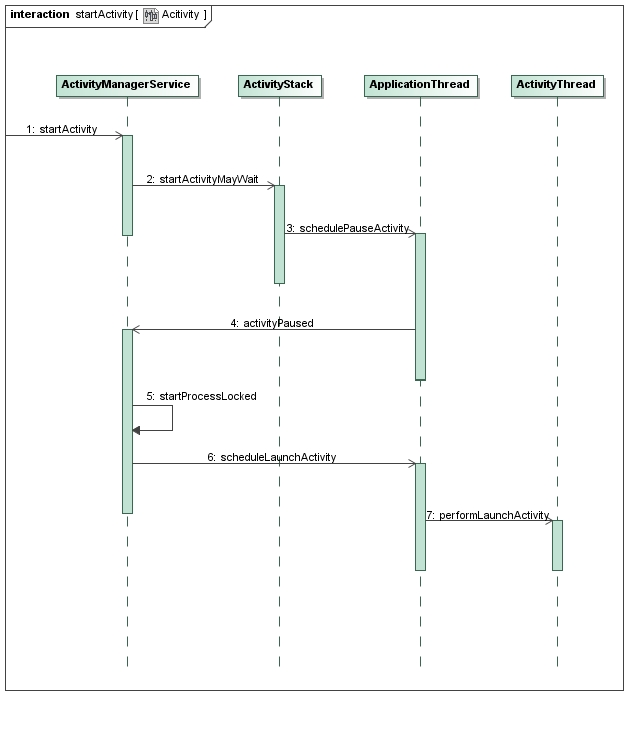
参考：<https://blog.csdn.net/Luoshengyang/article/details/6689748>

<https://blog.csdn.net/Luoshengyang/article/details/6703247>

<https://blog.csdn.net/lmq121210/article/details/82694781>

<https://blog.csdn.net/lmq121210/article/details/82705952>

1. 时序图



1. 有两种操作会引发Activity的启动，一：用户点击应用程序图标时，Launcher会为我们启动应用程序的主Activity；二、应用程序的默认Activity启动起来后，它又可以在内部通过调用startActvity接口启动新的Activity
2. Activity的启动方式有两种，一是显式的调用startActivity()方法；二是隐式的，隐式启动可以使得Activity之间的藕合性更加松散。
3. ActivityManagerService负责启动、管理四大组件和应用程序。
4. ActivityManagerService和ActivityStack位于同一个进程中，而ApplicationThread和ActivityThread位于另一个进程中；
5. ActivityManagerService还借助ActivityStack是来把所有的Activity按照后进先出的顺序放在一个堆栈中；
6. 一个ActivityThread来表示应用程序的主进程，而每一个ActivityThread都包含有一个ApplicationThread实例，它是一个Binder对象，负责和其它进程进行通信。
7. Activity可以通过配置android:process属性，使其运行在自己的进程中。android:process属性值以":"开头，则表示这个进程是私有的；以小写字母开头，则表示这是一个全局进程，允许其它应用程序组件也在这个进程中运行。
8. 整个应用程序的启动过程要执行很多步骤，但是整体来看，主要分为以下五个阶段：
9. Launcher通过Binder进程间通信机制通知ActivityManagerService，它要启动一个Activity，ActivityManagerService调用ActivityStack.startActivityMayWait获取准备要启动的Activity的相关信息；
10. ActivityManagerService通过Binder进程间通信机制通知Launcher进入Paused状态；
11. Launcher通过Binder进程间通信机制通知ActivityManagerService，它已经准备就绪进入Paused状态，于是ActivityStack通知ApplicationThread要进行Activity启动调度了，并且调用ActivityManagerService.activityPaused()判断是否需要创建一个新的进程，用来启动一个ActivityThread实例，即将要启动的Activity就是在这个ActivityThread实例中运行；（通过点击应用程序图标来启动Activity的情景，ActivityManagerService会调用startProcessLocked来创建一个新的进程，而对于通过在Activity内部调用startActivity来启动新的Activity来说，通过process属性值和uid在系统查找是否已有相应的进程存在，如果存在相应的进程，新的Activity就在原来的Activity所在的进程中进行启动，否则创建一个新的进程，然后在新进程中启动这个Activity）
12. ActivityThread通过Binder进程间通信机制将一个ApplicationThread类型的Binder对象传递给ActivityManagerService，以便以后ActivityManagerService能够通过这个Binder对象和它进行通信。
13. ActivityManagerService通过Binder进程间通信机制调用ApplicationThread.scheduleLaunchActivity方法，并通知ActivityThread现在一切准备就绪，可以执行Activity的启动操作了；
14. ActivityThread#handlerLaunchActivity()->ActivityThread#performLaunchActivity()->Instrumentation#newActivity()，Instrumentation#newActivity()通过ClassLoader反射创建Activity，如果判读Activity未被finish，则调用Activity#performStart()。
15. ActivityThread#handlerLaunchActivity()->ActivityThread#handlerResumeActivity()->ActivityThread#performResumeActivity()（判断Activity未被finish则继续后面流程）->Activity#performResume()。
16. ActivityThread#performResumeActivity()-> windowManagerImpl#addView() ->WindowManagerGlobal#addView()-> ViewRootImpl#setView() ->ViewRootImpl#requestLayout() ->ViewRootImpl#scheduleTraversals() ->Choreographer#postCallback()将绘制任务封装到runnable任务中 ->Choreographer#scheduleFrameLocked()向屏幕底层注册vsync信号的监听器 ->屏幕刷新信号vsync发出时会回调Choreographer 的onVsync() （同时注册下一帧屏幕刷新信号vsync）->Choreographer#doFrame()将绘制任务的runnable取出，并执行其run方法 ->ViewRootImpl#doTraversal() ->ViewRootImpl#performTraversals() ->ViewRootImpl#performMeasure()、performLayout()和performDraw()
17. 应用程序内部XActivity启动YActivity的流程和第9点一样，只需要把第9点的Launcher替换成XActivity。